ディープニューラルネットワーク実装編

B3 勉強会

2017.11.15

ニューラルネットワークの実装方法

- ニューラルネットワークを実装するためには様々な要素技術を実装しなければならない
  - 例．ネットワーク構造（層間の接続や損失関数）、関数微分（バックプロゲーション）、最適化手法（最急降下法など），学習データ構造
- これら実装を簡単に構築するためのライブラリが数多く出ている
- 殆どのライブラリは**各層のテンプレート**や**自動微分機能**、**最適化機能**等を備え、ネットワーク構造とデータを与えるだけで自動的に最適化してくれる

代表的なニューラルネットワーク関係のライブラリ

- Chainer（PFN）
- Tensorflow（Google）
- Keras（Google）
- CNTK（Microsoft）
- (py)Torch（Facebook)
- Theano（モントリオール大学）
  - 注）ほとんどのライブラリはpythonのインタフェースを持っている

- 今回は **chainer** を用いて手書き文字の認識を行ってみます

Github のインストール

GitHub（ギットハブ）はソフトウェア開発プロジェクトのための
共有ウェブサービスであり、Gitバージョン管理システムを使用する。
ウィキペディア

- B3ゼミではサンプルプログラムや課題の配布に利用する
- インストール方法は下記等参照
  http://www.atmarkit.co.jp/ait/articles/1603/31/news026.html#01

サンプルプログラム配布ページ

- https://github.com/NaohiroTawara/B3_seminor/tree/master/DNN

- Gitを使えば下記でまとめてダウンロードできます

```
$ git clone https://github.com/NaohiroTawara/B3_seminor
```

課題（結果の提出は求めません）

1. サンプルプログラムを回して手書き数字認識用ニューラルネットワークを構築してみよう

```
$ python train_mnist.py
```

2. 作成したニューラルネットワークを可視化してみよう

```
$ jupyter notebook analyze_mnist.ipynb
```

3. ニューラルネットワークの構造（**層数やユニット数、ミニバッチサイズなど**）を変えて学習したときに結果（**認識精度や計算時間など**）がどのように変わるか見てみよう

手書き文字（MNIST）

- 0～9の手書き文字
- データ数
  - 学習用：60000サンプル
  - テスト用： 10000サンプル
- データフォーマット
  - 28 × 28 pixel
  - 0-255の整数値からなるグレイスケール

実行時の注意

- 初回時のみ下記のコマンドにより必要なライブラリをインストールする必要があります

```
$ pip install chainer matplotlib sklearn numpy pydotplus Image
```

- 環境によっては他にもライブラリが必要な可能性があるので、もし下記のエラーが出たら適宜インストールしてください

```
ImportError: No module named <ライブラリ名>
```

実行例

初回時のみデータセットのダウンロードを行う

経過時間

各epoch時の学習・評価データの認識精度

各epoch（各データを１回ずつ入力した状態）時の学習・評価データの損失関数の値